Evaluating Congestion Control With Split TCP Performance Enhancing Proxies

Apoorv Joshi, William Alexander, Eesha Mulky, Ira Ramtirth, Karan Singh Dagar, Haris Mahmood Ansari

Department of Computer Science, North Carolina State University
Raleigh, NC, U.S.A

***Abstract*—Traditional TCP Congestion Control algorithms were designed with the typical wired networks available at the time in mind. As a result, deficiencies and performance issues exist when deployed in wireless, satellite, and other atypical networks. Split TCP Performance Enhancing Proxies have been used to improve performance in the past, but little research in the area exists in recent years. New TCP algorithms have been devised and new topologies (e.g. data centers) now exist, and it is unclear if PEPs are still advantageous when used in these new environments. In this paper, we model the performance of Split TCP PEPs in conjunction with modern topologies and algorithms, using the NS2 simulator. We find that these techniques can still be beneficial, though improvements are not universal. We present our results and provide recommendations on when to best use split connection PEPs, as well as which TCP variant is most suited for each topology.**

I. INTRODUCTION

Shortly after the development of the Internet, congestion control was added to TCP implementations in order to throttle senders when a level of congestion was detected in the network.[1] Many TCP congestion control algorithms exist, including TCP Tahoe, Reno, CUBIC and others. These algorithms allow a sender to regulate its own sending rate by detecting suspected congestion in the network after observing either a timed out response or triple duplicate ack. This design was chosen because of the "end-to-end argument" principles which guided the design of the Internet, stating that anything which could be implemented on the endpoints should be implemented there.[3] While this is the decision chosen by TCP/IP, this is not the only way to implement congestion control or always the best way.

TCP congestion control mechanisms were tuned for the traditional wired model of networking that existed at the time (small, local, wired networks), in which dropped packets are typically caused by congestion. As a result, the traditional model infers congestion by assuming that all packet losses are congestion related.[1] This is not always the case, especially in modern systems. Wireless networks, for instance, can encounter lost packets due to wireless signal issues and bit errors, inadvertently triggering congestion control algorithms that will rate limit a sender unnecessarily.[3][5][8] Because networks have grown from small LANs to large continental services, links with high Bandwidth-Delay products ("long fat pipes") also suffer, due to the long feedback loop required to receive ACKs and quickly calibrate the sending rate.[2][3][7] These concerns have been shown to affect performance in the areas of wireless networks, satellite links, and other networks over the years.[5][6][7][8] As networking evolves, other issues may arise in newer network topologies, such as data centers, content delivery networks, software defined networks and others. It is not always clear if the original model is the best method for congestion control in these networks either.

Systems have been developed to enhance performance and mitigate the problems in existing algorithms. Many of these systems do not strictly follow the model of traditional TCP/IP.[2][3] There are multiple solutions which can be implemented on nodes within the network itself and avoid relying solely on the endpoints for congestion detection and control. Virtual circuit networks detect and control congestion directly within the network, but direct network layer congestion control is not appropriate in the global Internet.[2][3] We can achieve benefits by having the endpoints still perform the congestion control, but also utilize help from points within the network for detection. One such way is through Explicit Congestion Notification bits, which are set by routers with queue management systems that monitor their own congestion levels.[2][8] These routers send a type of warning signal back to senders to indicate that congestion may be coming and to preemptively lower their window. While this is an improvement on current design, after decades of development it is still little supported or deployed in the Internet.[6] Further, while it may help in long delay networks, it does not help with the problem of performance issues in links with high error rates, such as wireless.

A more promising technique, which allows endpoints to still perform congestion control but helps them achieve better results, is to use Performance Enhancing Proxies (PEPs), such as split connection TCP.[2][3][7][12] Splitting the TCP connection at multiple hops along a path allows for triggering congestion avoidance closer to the sender, allowing for a faster feedback loop for calibrating windows.[3][7] Further, splitting the TCP connection can help to isolate the congestion control to certain parts of the network.[3] This technique has been used in satellite communication, which suffers from long delay issues when calibrating congestion windows.[3][7] Further it has been used in wireless networks to help mitigate transmission errors. A similar example is the SNOOP protocol, which works at the data link level to "snoop" on TCP packets and monitor connection issues. It hides and creates ACKs at

wireless base stations, to mask wireless losses from senders so that they do not interpret them as congestion losses.[3][4][8] While SNOOP provides a large performance increase over traditional TCP, at least one study has shown pure split connection techniques provide a significant improvement over techniques like SNOOP that do not completely split the connection.[12] Regardless, such proxy methods are shown to be more reliable in terms of stopping congestion before it has a chance to result in a collapse of the network.[4][5] They are able to help senders calibrate their congestion control activities in a more timely manner, by being closer and not requiring a full RTT wait for an ACK from the destination receiver. This means a quicker adjustment both up or down, of the congestion window and reaching optimal send rate sooner.[4][5][7][8][12] Further, this type of technique benefits from the existing end-to-end behavior of endpoints (still uses ACK inference) and can be implemented transparently within the network. [3][4][12]

Before attempting to use any of these techniques, we must be mindful that we are using them effectively and without adverse effects. In RFC 6077, "Open Issues In Congestion Control", network supported techniques are listed as the "most critical" of issues, as getting them right is important in order to ensure we are getting a high level of congestion notification as well as not breaking the end to end model of the internet, which supports heterogeneous networks.[2] In concurrence, RFC 3135, "PEPs To Mitigate Link Related Degradations", states that split connections and other PEPs should only be used in those situations where there is a clear benefit, not in the general case, as the risk associated with breaking the benefits of the end-to-end model exists.[3]

Since it is clear that split connection congestion control techniques have certain benefits, they should be used in scenarios only where they provide a clear improvement since there is certain risk involved with it. In the past, most major work in this area has focused on wireless and satellite networks, without much focus on modern types of networks such as data center communication and other newer topologies. Also lacking is research on how split connection PEPs work when used in conjunction with modern TCP algorithms, some of which (e.g., CUBIC) are focused on traditional use cases for PEPs (long delay networks). It would be useful to see if such modern topologies and modern TCP algorithms would still benefit from split connection techniques and whether they are worth deploying in those environments. As such, in this paper we explore split connection TCP for congestion control and measure its performance in various network simulations. We use NS2, an open source networking simulation tool, for modeling various network topologies which may or may not benefit from split connection PEPs. We compare these results to gain an understanding of the possible improvements and to answer whether they are worth pursuing for future implementations. As we will show in our results, we find that split connection TCP proxies can be beneficial in many networks, though that benefit is dependent on the TCP variants used for each topology. By presenting our findings, network implementers may use this data to determine when to deploy split connections as well as the configurations that will work best for their networks.

The paper is organized as follows: in Section 2, an overview of the existing literature in the field of congestion control techniques and split TCP proxies are reviewed and compared. In Section 3, we discuss a number of topologies in which we believe split connection proxies may provide an improvement. In Section 4, we provide an overview of our experimental setup and implementation within the NS2 simulator and discuss our results. In Section 5, we provide ideas for future work, before concluding the paper in Section 6.

II. RELATED WORKS

As we seek to answer the problem of TCP congestion control in a wireless network we studied and analyzed previously written papers that help motivate us to define the problem unambiguously. It became clear that TCP responds to all packet losses by invoking congestion control and avoidance algorithms, resulting in degraded end-to-end performance in wireless systems.[9][14] We explored schemes that are designed to improve the performance of TCP. We preferred to choose Split TCP connection to evaluate and analyze the performance of our experimental setup. In our paper, we evaluate Split Connection approach via simulating different scenarios on NS2 that proves that PEP improves congestion control mechanism induced by TCP.

It is interesting to note that an alternative solution to congestion control problem was also suggested as E-TCP, which improves upon conventional TCP congestion and flow control in a wireless environment. The authors explore that providing TCP with an explicit acknowledgment about the cause of packet loss via ACKC-CLN (congestion-corruption loss notification) can help.[10] If only a small amount of additional functionality is introduced at the base station and sender, proposed transport protocol could maintain large congestion window size, more packets could be sent to the receiver and throughput could be increased. However, it was seen that loss of an acknowledgement packet could adversely affect the model. Moreover, E-TCP was not tested on handoff related issues. This shows that the functionality of this design relies heavily on the assumption that an ACK-CLN should not be dropped. However, since probability of it happening is high, we believe that introduction of PEPs such as Split Connection would actually be a better approach to addressing the same problem.

Motion across wireless cell boundaries causes increased delays and packet losses. TCP interprets such delays and losses as a sign of network congestion and throttles their transmissions. It was found that TCP's retransmission timeouts cause pauses in communication that last 0.8s after each cell crossing. Moreover, it is analyzed that an end to end fast retransmission scheme can reduce such pauses to levels that are more suitable to human interactions.[11] We further explore the practical implications of TCP congestion control to the end user and although the proposed approach explores only losses caused in mobile systems when changing cells, we understand that such a system when deployed with Split Connection TCP will show further enhancement.

Split Connection TCP employs a proxy at the boundary of wired networks and wireless networks and although this technique is advantageous, we still have the issue of maintaining the end-to-end semantics. To address the end-to-end semantics problem, a novel technique called Semi-Split TCP is explored. In Semi Split TCP, two agents are responsible to interact with the sender and the mobile host respectively, a Semi Split receiver and Semi-Split sender. The agents "hook" the TCP packets from network layer. Semi-Split receiver agent buffers TCP data and properly acknowledges the sender using spoofed ACK. Semi-Split sender agent forwards the buffered data to the mobile host by using enhanced transport layer protocol.[12] It was observed that Semi Split TCP has performance similar to Split TCP and it would be better to use Semi Split TCP because it also maintains end-to-end semantics. As we explore split connection performance in our simulation, we also acknowledge the concern of end-to-end semantics and believe that Semi Split TCP is the answer for those issues

TCP algorithms which seek to fix the issues with long delay RTTs for congestion control calculations have been suggested such as BIC.[2] We have seen the implementation of SNOOP Protocol in wireless networks to improve TCP performance.

Snoop modifies the network-layer software at the proxy and maintains end-to-end semantics.[4] A further enhancement to SNOOP which included the ability to duplicate fake ACKs by modifying the SNOOP procedure was done by Moojid, et al.[5]

ECN bits can be used to indicate congestion in routers and improve enforcing congestion control, while SNOOP can be used to suppress incorrect congestion control notifications through TCP ACKs. Combining these two algorithms will save the retransmit time.[8] PEPs (Performance Enhancing Proxies) are used in networks having links with high bandwidth-delay product. The analysis of split TCP connection using web caches in high delay satellite links to retrieve HTTP content.[7]

In order to understand the dynamics and stability of split connection used for congestion control in an end-to-end TCP connection, there are some mathematical equations which are derived and proved in some scenarios by the authors of the paper [13]. They emphasize on the buffer size of proxy on the total throughput gain in case of splitting the TCP connection. They introduced a "backpressure mechanism" which is used to avoid packet loss in proxies. A backpressure mechanism is proposed to limit the sending rate to the forwarding rate in presence of saturated proxy buffer which ultimately prevent buffer overflows.[13]

III. MODEL TOPOLOGIES

Prior research demonstrates several scenarios where PEPs offer increased performance. It also pointed to the fact that the amount of research about performance of PEPs in modern networks and modern TCP variants was less. Here we have identified a number of topologies which are deserving of experimentation. For each of these types of networks we evaluate performance with and without a split connection proxy and use several TCP variants on the senders. We focus

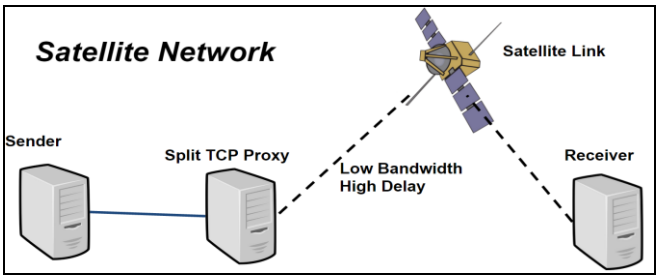


Figure 1: A Low Bandwidth High Delay satellite network with split TCP proxy

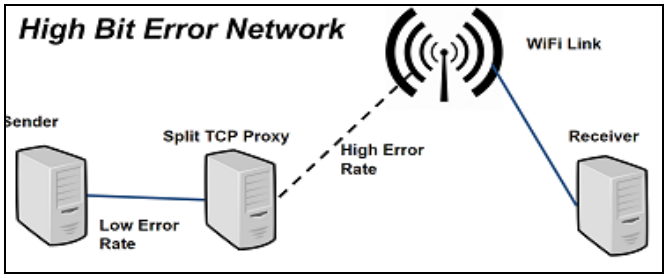


Figure 2: Network with high error and drop rate on wireless link with split TCP proxy

here on just the simple case of a TCP sender/receiver node at the proxy which forwards packets, without any optimized PEPs as described in some prior research. By focusing on the trivial implementation, we can control for variations in split connection proxies and obtain a base case for performance evaluation. As an illustration, several candidate topologies will be used as our basis for research.

In Figure 1, we see a satellite connection which includes a low bandwidth and high delay link between a sender and receiver, which would result in a long feedback delay for ACK arrival, reducing performance. If congestion occurs at one end of the network, it will take much longer for the other end of the network to control it. Here we introduce a split connection at what would be the ground station receiver, as our split connection proxy node.

In Figure 2, it shows a network which includes a high bit error link such as a wireless network which may have radio interference. Our proxy in this scenario is implemented as would be found at a wireless base station, interfacing with the wired network.

In Figure 3, we see a cross country high speed datacenter to datacenter WAN which includes a high bandwidth-delay product link. Such a link would result in a long feedback delay for ACK arrival, and take a long time to reach optimized window size. Modern variants of TCP such as CUBIC seek to improve these situations. In our proxy scenario, we place the proxy as a waypoint between the two cross country links.

In Figure 4, we see a scenario which is likely to see less benefit from split connection TCP. However, this is not confirmed, as prior research was conducted before the prevalence of data centers and therefore literature is not available for these types of topologies. However, benefits of

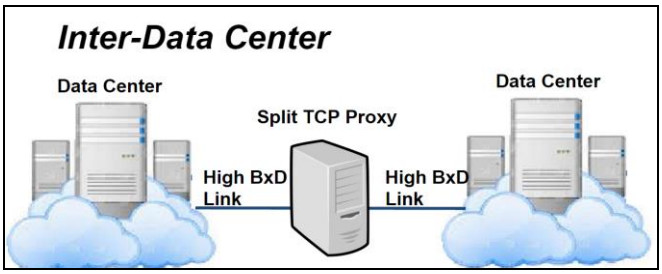


Figure 3: Cross country datacenter to datacenter high bandwidth-delay product links

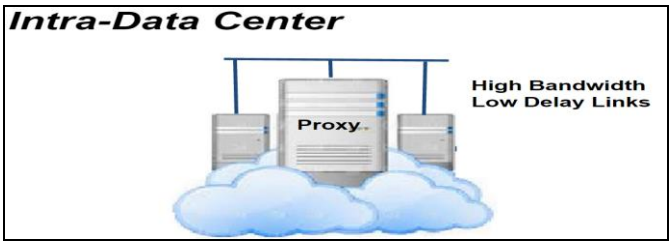


Figure 4: High Bandwidth Low Delay network within a data center

quick feedback loops will not be as beneficial in these networks, since a low delay and high bandwidth is already available for rapid sizing of congestion windows. There is still a potential for improvement, however, if split connections could more quickly control congestion within the network and stop it from spreading. It is unclear if the overhead of the additional connections will negate any benefits.

These network models give a lower bound by which performance can be measured and can be extrapolated to evaluate other types of proxies. Intrusion detection systems, firewalls, and web caches all perform some proxy functions which often split TCP connections transparently. Similar performance increases or decreases we see in our model topologies should also be found in these real world proxies as well.

IV. SIMULATION

IV.I Performance Metrics

The majority of our work on split TCP is experimental evaluation of the throughput performance of the network by splitting connection technique. In addition to the throughput metric, our work is focused on the analysis of buffer occupancy in the proxy node and on packet delays. We have graphically analyzed how the congestion window size of TCP sender as well as bandwidth requirement changes when we observe dropping of packets at proxy buffer due to high congestion.

IV.II TCP Variants

We performed a comparative analysis of the following TCP variants in different model topologies:

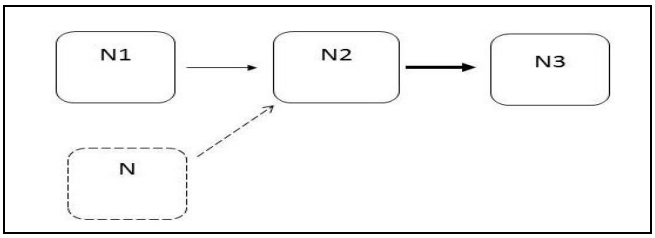


Figure 5: Base case scenario (TCP end-to-end)

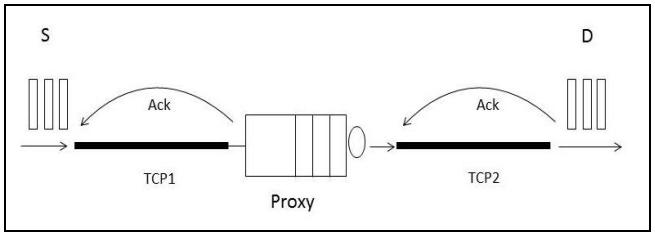


Figure 6: Split TCP network scenario

- TCP New Reno: It implements the Additive Increase Multiplicative Decrease algorithm for adjusting the size of the congestion window based on whether packet was acknowledged or lost. It calculates the window size based on the phase in which it is that is either Slow Start(SS), Congestion Avoidance(CA), Fast Retransmit(FA) or Fast Recovery(FA).
- TCP Vegas: It adjusts the congestion window by considering the difference between the actual throughput and maximum achievable throughput so that the number of packets in the router buffers is less.
- TCP Cubic: This variant updates its window size independent of RTT which leads to a more balanced bandwidth distribution among the users. It uses a cubic function in place of the normal linear growth function in standard TCP for setting the congestion window (cwnd) size.

These TCP variants initiate the congestion control algorithms on different parameters. TCP New Reno and Vegas were the ideal choice for studying the effect of split TCP on congestion window. On the other hand, Cubic would help in studying the effects of split TCP on bandwidth more accurately.

IV.III Experimental Setup

We implemented a base case topology which consists of three nodes and two dedicated links between N1-N2, and N2-N3 in NS2 simulator as shown in figure 5. NS2 simulator provides substantial support for simulation of TCP, routing protocols in a wired or wireless network virtually. We configured end-to-end TCP connection between N1 and N3, where N1 and N3 are sender and receiver respectively. N2 is behaving like a hub which is forwarding the packet from one end to the other. We tested the scenario by configuring the three TCP variants: TCP New Reno, TCP Vegas, and TCP Cubic. Random traffic generation is implemented in two ways

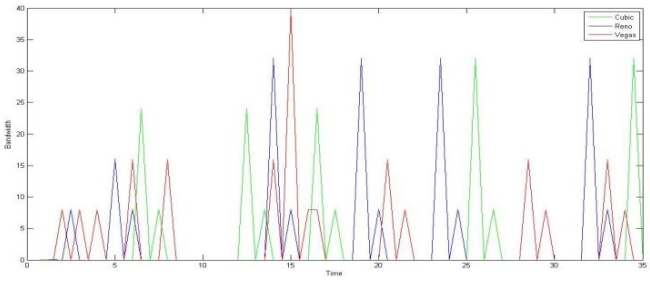


Figure 7: Bandwidth Vs time in full TCP end-to-end

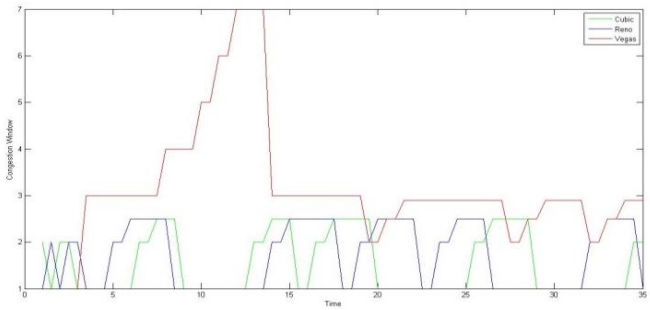


Figure 8: Congestion Window Vs time in full TCP end-to-end

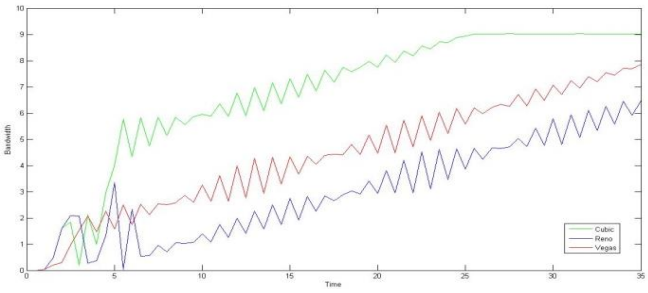
in order to create congestion in N2 node. Firstly, ftp is running over the TCP connection from sender to receiver and secondly, UDP traffic is generated from an arbitrary node N.

In order to split the end-to-end TCP connection, we modified the above topology. We have a traffic source S that sends packets to D through two long-lived TCP connections TCP1 and TCP2 in cascade as shown in figure 6. In this scenario, we showed the possibility of heavy tail asymptotic for proxy buffer occupancy where we considered proxy as the middle node.

In our simulation setup, we mainly focus on the study of buffer occupancy in the proxy and end-to-end delay of packets by having various link configurations in order to implement the model topologies described in Section 3. Further, we graphically analyze our result and give guidance to when and when not to use split connection TCP for congestion control.

IV.IV Simulation Result

We graphically analyzed our model topologies by plotting graphs of the congestion window (in Bytes) of the TCP sender against time (in seconds). We further provide graphical analysis of the minimum bandwidth required (in Mbps) for sending the data from sender to receiver. We ran the simulation for each scenario multiple times to get an average range of values by monitoring each scenario for different instances of time as per link configuration. The following figures 7 and 8 are the graphical analysis of our base case scenario, a typical LAN link, having full TCP end-to-end in which we have the link configuration as 10Mbps, 50ms, and 0% drop rate on both dedicated links. The monitoring time was 35s.



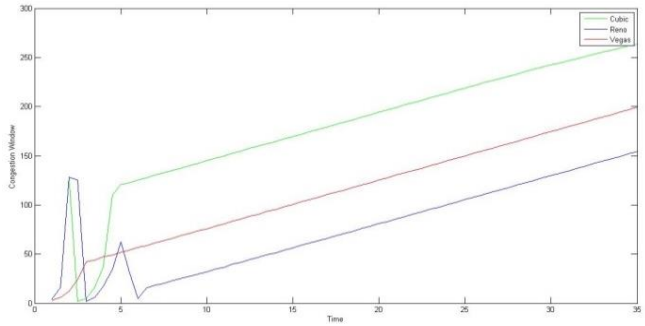


Figure 9: Analysis of Split TCP connection of base case topology

After implementing splitting technique on our base case, the results were different as shown in figure 9. We can observe a quick increase on reaching the maximum window size after 10s.We found that split connection in a typical LAN, especially in case of Cubic, gives a tremendous increase in performance during aggregate interval of 5 to 35s.

The graphical analysis for our model topologies are as follows:
(i) Satellite Network with low bandwidth and high delay

Link Configuration: 5Mbps, 1000ms, 0% error rate
In our simulation, we are constantly generating traffic through UDP from an arbitrary node so that there is congestion on the link connection from the proxy to the receiver. The analyses are shown in figures 10 and 11. We monitored the simulation for 50s, to account for high delay links in order to plot the graph. We found in the satellite case that split TCP connections perform best when used in conjunction with Cubic TCP during aggregate interval of 15 to 49s. We can see growing window size after 15s in case of Cubic. In particular, Vegas perform poorly with split in this scenario after a quickly growing congestion window.

(ii) Wireless Network with high bit error rate

Link Configuration: 10Mbps, 100ms, 5% error rate
In this test, we configured a lossy link between the proxy and the receiver with a 0.5 probability of dropping a packet. We monitored it for 20s. The results are shown in figures 12 and 13.

Performance in this topology seems to be opposite to the satellite case. In this scenario, we see Cubic bandwidth

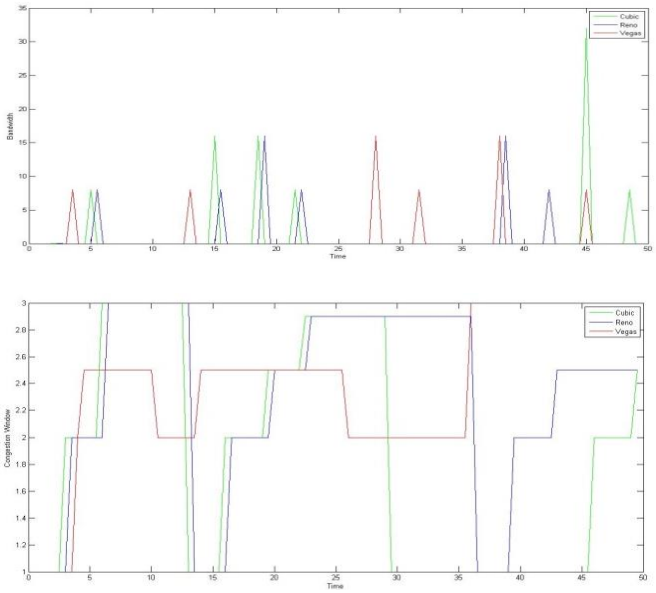


Figure 10: Analysis of Non-Split TCP connection- Satellite network

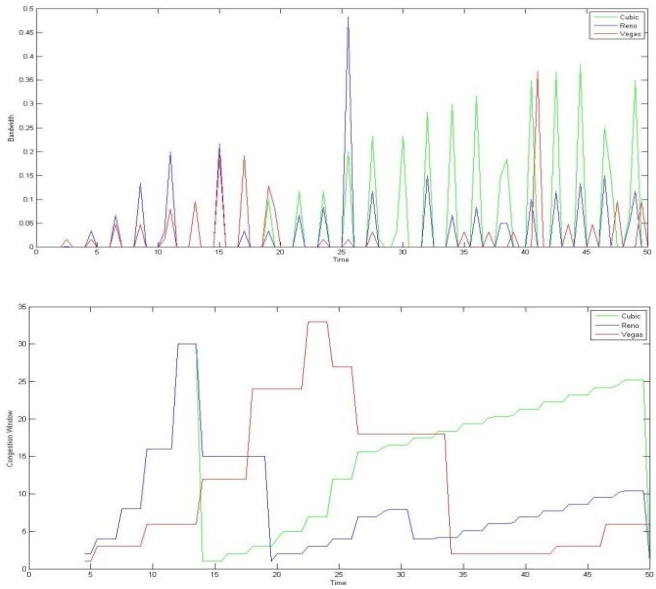


Figure 11: Analysis of Split TCP connection of Satellite network

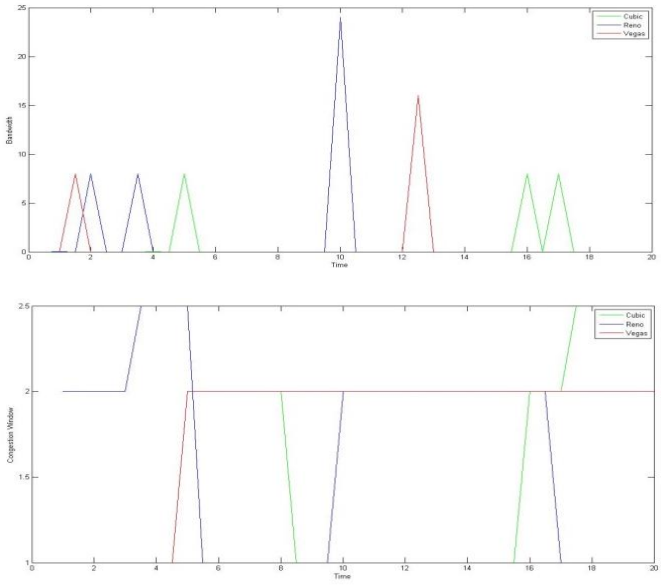


Figure 12: Analysis of Non-Split TCP connection of Wireless network

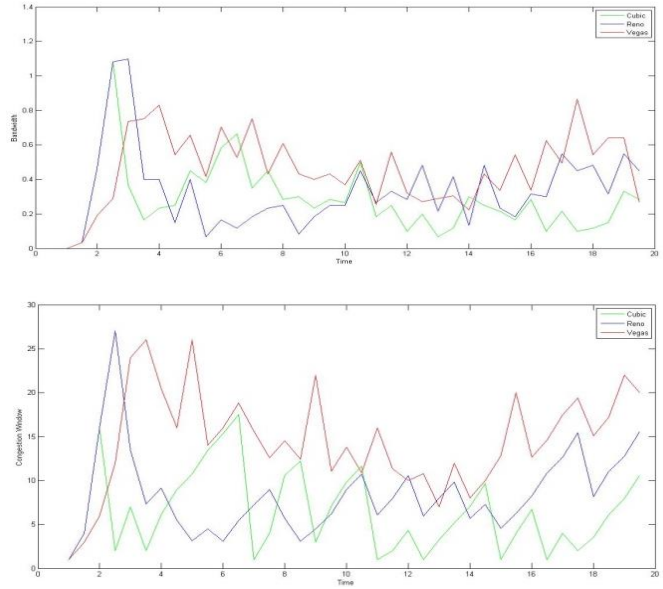


Figure 13: Analysis of Split TCP connection of Wireless network

performance is worse during drops. Split TCP is able to improve the performance of New Reno and Vegas, yet Cubic is more affected by drops in this scenario.

(iii) Datacenter Network with high bandwidth-delay product link

Link Configuration: 1000Mbps, 1000ms, 0% drop rate
This test represents a long haul high bandwidth network, such as those between data centers. We first implemented and analyzed the performance of a full end-to-end TCP connection in this scenario, with links having high bandwidth delay product.

We compared its performance to that of a split connection at the proxy and monitored the simulation for 100s, as shown in figures 14 and 15.

The results here were surprising in the case of Cubic which achieved a lower performance as compared to Reno and Vegas, which was not expected. However, on average, Cubic is better in achieving high bandwidth at 75s. In this case, after getting the average results for both split and non-split, we found that splitting the connection did not affect the performance. New Reno was much better at maintaining a large congestion window and achieving a higher bandwidth.

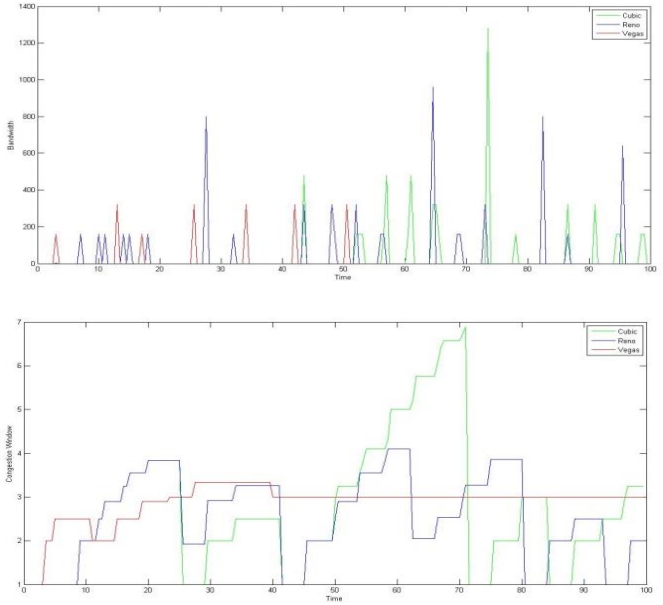


Figure 14: Analysis of Non-Split TCP connection of inter-datacenter network

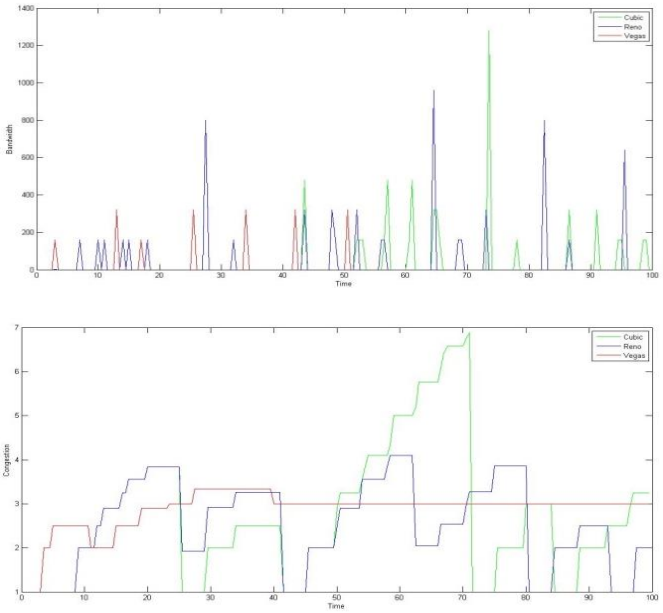


Figure 15: Analysis of Split TCP connection of inter-datacenter network

(iv) Datacenter Network with high bandwidth and low delay link

Link Configuration: 1000Mbps, 10ms, 0% drop rate
In this test, we implemented a network as may exist in a data center, where large capacity links are located nearby within the same building. Figure 16 shows the results without using a split connection, against the results using a split TCP in figure 17, which were monitored for 40s.

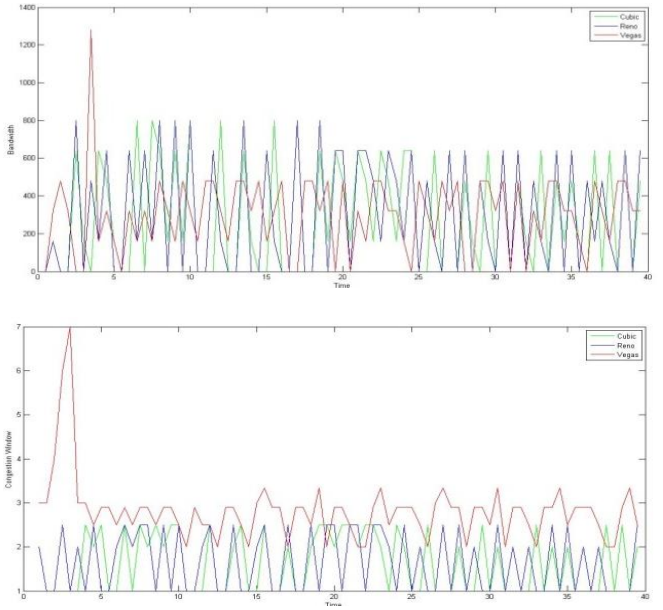


Figure 16: Analysis of Non-Split TCP connection in intra-datacenter network

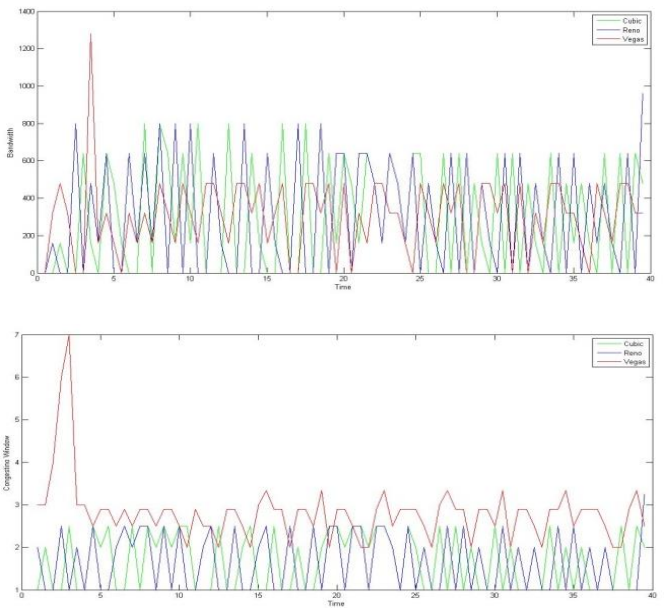


Figure 17: Analysis of Split TCP connection of intra-datacenter network

It is apparent in this case that there was no benefit in using the split connection TCP. This is to be expected, as in the case of a large bandwidth and low delay network, any positive effects of the proxy will not be found, as the time to ACK and scale the windows will be nearly identical. This is confirmed by the simulation results.

V. FUTURE WORK

A number of extensions could be done to this work to get a fuller picture of the use cases of split connection TCP performance. In particular, we have only tested against three TCP variants here, while many more exist. We tested against popular algorithms which are widely deployed, though algorithms tailored to specific use cases exist. Data Center TCP (DCTCP) would be a useful algorithm to test in data center configurations. TCP Westwood is a modification of New Reno which would be beneficial for testing in long delay networks. Further, TCP Hybla is primarily designed for use in satellite networks and could be beneficial in those simulations. All of these should also be compared in split TCP networks.

More future work could be done with split TCP while using other types of enhancements to TCP, which may not be congestion control related. One such enhancement is Semi-Split TCP, which can be used to make split TCP behave in a more end-to-end friendly manner, while only slightly reducing performance. [12] Another is TCP Fast Open, which speeds up the TCP handshaking procedure. Fast Open could help to reduce some overhead introduced by opening another TCP connection at the proxy.

Finally, as simulations cannot perfectly model all variables, the work here could benefit from running these scenarios over real networks in order to get actual experimental results on real hardware.

VI. CONCLUSION

The results of our simulations show that the justification for using split TCP proxies depends greatly on which type of network you are implementing it in. In smaller networks with low delay (e.g., within datacenters) we notice no benefit and possibly a detriment to its use. We do see improvements in other types of networks, but the improvements found vary depending on the type of TCP variant used. CUBIC for example performs well in split TCP satellite networks, but poorly in wireless networks with high drop rate. In the case of New Reno and Vegas we notice the exact opposite result. We hope additional work is contributed in this area using a larger number of TCP variants in order to get a clearer view of when these benefits are present.

We have shown that the benefits of TCP performance enhancing proxies are highly dependent on the topology and variant with which is it used. As our study has confirmed, there is no single best variant and the network designer must keep this in mind as they decide whether to implement split connection proxies.

REFERENCES

[1] S. Floyd. Network Working Group, September 2000. RFC2914. “Congestion Control Principles”

[2] D. Papadimitriou, M. Welzl, M. Scharf, and B. Briscoe. Network Working Group, February 2011. RFC6077. “Open Issues in Internet Congestion Control”

[3] J. Border, M. Kojo, J. Griner, G. Montenegro, and Z. Shelby. Network Working Group, June 2001. RFC3135. “Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations”

[4] H. Balakrishnan, S. Seshan, E. Amir, and R. Katz. 1st ACM Conference on Mobile Computing and Networking (Mobicom), November 1995. “Improving TCP/IP Performance over Wireless Networks”

[5] F. Moojid, M. Habaebi, and B. Ali. JURUTERA, July 2005. “Improving Wireless Snoop Performance Using Fake ACK Technique”

[6] M. Kuhlwind, S. Neuner, and B. Trammel. Proceedings of the 14th International Conference on Passive and Active Measurement, October 2013. “On the State of ECN and TCP Options on the Internet”

[7] N. Ehsan, M. Liu, and R. Ragland. International Journal of Communication Systems, January 2003. “Evaluation of Performance Enhancing Proxies in Internet over Satellite”

[8] M. Chawhan and A. Kapur. International Journal of Computer Applications, August 2010. “Performance Enhancement of TCP Using ECN and Snoop Protocol for Wi-Max Network”

[9] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz. ACM SIGCOMM, August 1996. “A Comparison of Mechanisms for Improving TCP Performance over Wireless Links”

[10] D. Chandra, R. Harris, and N. Shenoy. 3rd IASTED International Conference on Wireless and Optical Communication, July 2003. “TCP performance for future IP-based networks”

[11] [11] R. Caceres an L. Iftode. IEEE Journal on Selected Areas in Communcation, June 1995. “Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments”

[12] F. Xie, N. Jiang, Y. Ho, and K. Hua. IEEE Conference on Local Computer Networks, October 2007. “Semi-Split TCP: Maintaining End-to-End Semantics for Split TCP”

[13] Francois Baccelli, Giovanna Carofiglio, Serguei Foss. Cambrigde University Press, 2009. “Proxy caching in split TCP: dynamics, stability, and tail asymptotics”

[14] Nedo Celandroni, Francesco Potortì, International Journal of Communication Systems, Vol. 16, No.1, pp. 63-79, February 2003, “Maximizing single connection TCP goodput by trading bandwidth for BER”